ถุงมือสำหรับตรวจจับท่าทางมือ

SMART GLOVE FOR GESTURE RECOGNITION

พัทธวีร์ ชุมภูวร

พิทวัส คุณกะมุต

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2561

บทคัดย่อ

ABSTRACT

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญรูป

รูป **หน้า**

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหา

ในปัจจุบันมีการพัฒนาอุปกรณ์ต่าง ๆ โดยมีเป้าหมายเพื่อให้ผู้ใช้สามารถใช้งานอุปกรณ์ทางอิเล็กทรอนิกส์โดยใช้ร่างกายของตนเองเป็น Input ได้ และได้รับความรู้สึกสมจริง หรือสะดวกสบายมากขึ้นหากเทียบกับการใช้งาน Input ปกติอย่างเช่นคีย์บอร์ด, เมาส์ หรือจอสัมผัส

หากกล่าวถึงอุปกรณ์ที่พยายามตรวจจับท่าทางของมือ ในปัจจุบันก็มีการพัฒนาขึ้นมาหลากหลายประเภทเพื่อวัตถุประสงค์ใดอย่างหนึ่งอย่างชัดเจน ตัวอย่างเช่น ถุงมือตรวจจับท่าทางมือที่ใช้งานร่วมกับเทคโนโลยี Virtual Reality เพื่อใช้ในการเล่นเกมส์เพื่อความบันเทิง หรือฝึกฝนทักษะปฏิบัติเสมือนจริง และถุงมือตรวจจับท่าทางมือเพื่อใช้แปลภาษามือเป็นคำหรือตัวอักษรภาษาอังกฤษ เป็นต้น

รูป 1.1 ถุงมือตรวจจับท่าทางมือที่ใช้งานร่วมกับเทคโนโลยี Virtual Reality

รูป 1.2 ถุงมือตรวจจับท่าทางมือเพื่อใช้แปลภาษามือ

แต่เนื่องจากถุงมือตามตัวอย่างที่กล่าวมาข้างต้นเป็นถุงมือที่ถูกพัฒนาขึ้นมาเพื่อวัตถุประสงค์ใด ๆ อย่างชัดเจน ท่าทางต่าง ๆ ที่ใช้เป็น Input รวมถึง Output จึงจำกัดอยู่ในขอบเขตที่ผู้พัฒนาได้กำหนดไว้เพียงเท่านั้น ผู้ใช้จริงไม่สามารถกำหนดท่าทางต่าง ๆ โดยเฉพาะได้ หรือต้องมีการพัฒนาต่อโดยใช้ชุดพัฒนาซอฟต์แวร์ที่รองรับจากผู้พัฒนา ร่วมกับความรู้เฉพาะด้านเพียงเท่านั้น

ทางผู้พัฒนาจึงสนใจที่จะสร้างถุงมือเพื่อตรวจจับท่าทางมือ โดยใช้เทคโนโลยี Machine Learning ที่ได้รับการพัฒนาประสิทธิภาพขึ้นและมีความนิยมในปัจจุบัน เพื่อลดข้อจำกัดในด้านที่กล่าวมาข้างต้นลง โดยการแยกโครงสร้างการเคลื่อนไหวของท่าทางออกเป็นท่านิ่งหลาย ๆ ท่าที่เชื่อมต่อกัน และมีแอพพลิเคชั่นให้ผู้ใช้สามารถตั้งค่าชุดของท่าทางที่จะใช้เป็น Input และตั้งค่า Output ได้ ดังนั้นถุงมือจะสามารถตรวจจับท่าทางที่เป็นการเคลื่อนไหวได้หลากหลายแบบหลากหลายท่าติดต่อกันตามที่ต้องการ

1.2 วัตถุประสงค์ของโครงงาน

1.) พัฒนาถุงมือเพื่อการตรวจจับท่าทางมือรวมถึงระบบที่เกี่ยวข้อง
2.) ศึกษาการใช้งานไมโครคอนโทรลเลอร์และระบบปฏิบัติการ Linux ภายใน
3.) ศึกษาและพัฒนาเว็บแอพพลิเคชั่นและ Server เพื่อติดต่อกับ Hardware
4.) ศึกษาการประมวลผลข้อมูลของเซนเซอร์ผ่านกระบวนการทาง Machine Learning

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.) ถุงมือสามารถอำนวยความสะดวกให้กับผู้ใช้งานได้
2.) ถุงมือสามารถให้ความบันเทิงกับผู้ใช้งานได้
3.) ถุงมือสามารถช่วยเป็นสื่อกลางในการสื่อสารโดยภาษามือได้บางส่วน
4.) ผู้ใช้สามารถเข้าถึงและตั้งค่าการใช้งานถุงมือได้ง่าย

1.4 ขอบเขตของโครงงาน

การจับท่าทางของถุงมือที่เป็นท่านิ่ง จะสามารถทำได้อย่างน้อย 20 แบบ แต่ละแบบสามารถวางมือได้อย่างน้อย 5 แบบ สำหรับท่าที่เป็นเคลื่อนไหว สามารถทำได้อย่างน้อย 20 ท่า

และโดยรวมแล้ว จะสามารถตรวจจับท่าทางภาษามือที่เป็นตัวอักษรภาษาไทยได้อย่างน้อย 30 ตัว

1.5 ข้อจำกัดของโครงงาน

รูปแบบท่าทางของของมือผู้ใช้ที่ไม่เหมือนกันอาจจะส่งผลให้ความแม่นยำแตกต่างกัน เกิดจากค่าที่ได้จากเซนเซอร์ ซึ่งอาจจะมีค่าผิดพลาดหรือแตกต่างมากพอที่จะทำให้ตรวจจับได้ท่าที่ผิดพลาดได้ โดยเฉพาะเซนเซอร์ที่ใช้วัดความงอของนิ้วมือ ซึ่งวัดการงอได้เพียงแค่ทิศทางที่กำมือเท่านั้น สำหรับผู้ใช้ที่สามารถกางและงอนิ้วมือไปในทิศตรงข้ามได้มาก อาจจะทำให้เกิดการตรวจจับที่ผิดพลาดได้ แต่ท่าที่อยู่ในกรณีดังกล่าวนั้น ถือว่าเป็นส่วนน้อยของท่าทั้งหมดที่ถุงมือควรตรวจจับได้ และไม่ได้อยู่ในขอบเขตการพัฒนา

1.6 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

หัวข้อกิจกรรม	**เดือน**									
	ส.ค.	**ก.ย.**	**ต.ค.**	**พ.ย.**	**ธ.ค.**	**ม.ค.**	**ก.พ.**	**มี.ค.**	**เม.ย.**	**พ.ค.**
1. ค้นหาหัวข้อที่สนใจและปรึกษาหัวข้อดังกล่าวกับอาจารย์										
2. ศึกษาเทคโนโลยีที่ใช้พัฒนา										
2.1 ศึกษาวิธีการใช้งานระบบปฏิบัติการ										
2.2 ทดสอบการติดตั้งส่วนประกอบต่าง ๆ ที่ใช้ในการพัฒนา										
3. ออกแบบ										
3.1 ออกแบบโครงสร้างถุงมือ										
3.2 ออกแบบโครงสร้างระบบ										
3.3 ออกแบบโครงสร้างและวิธีประมวลผล Dataset										
3.4 ออกแบบเว็บแอพพลิเคชั่น										

หัวข้อกิจกรรม	***เดือน***									
	ส.ค.	**ก.ย.**	**ต.ค.**	**พ.ย.**	**ธ.ค.**	**ม.ค.**	**ก.พ.**	**มี.ค.**	**เม.ย.**	**พ.ค.**
4. พัฒนา										
4.1 ประกอบถุงมือเพื่อให้สามารถใช้งานได้เบื้องต้น										
4.2 พัฒนาเว็บแอพพลิเคชั่นเพื่อใช้แสดงผลและสร้าง Dataset										
4.5 พัฒนาเว็บแอพพลิเคชั่นเพื่อใช้ตั้งค่า Input/output										
5. ทดสอบ ปรับปรุง และแก้ไข										
5.1 ทดสอบและแก้ไขการทำงานโดยรวมของระบบ										
5.2 พัฒนาส่วนจ่ายไฟให้ถุงมือ										
5.3 ปรับปรุงรูปลักษณ์ถุงมือ										
5.4 ปรับปรุงเว็บแอพพลิเคชั่น										
5.5 ปรับปรุงโมเดล Machine Learning										

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

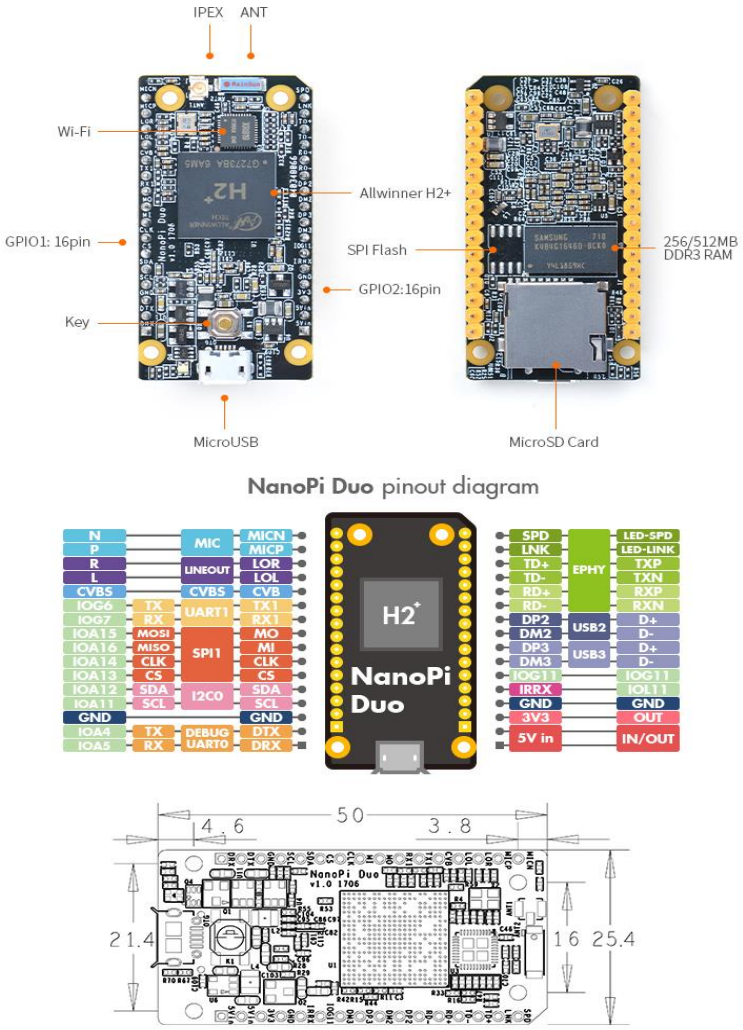
2.1 เทคโนโลยีด้านฮาร์ดแวร์

2.1.1 ไมโครคอนโทรลเลอร์ (Microcontroller)

Nanopi-DUO หน่วยประมวลผลขนาดเล็กที่สามารถวางไว้บนข้อมือได้อย่างสะดวก มีคอร์ในการประมวลผลถึง 4 คอร์ จึงสามารถทำงานหลายงานพร้อมกันได้สะดวก เหมาะสมกับโครงการที่จำเป็นจะต้องประมวลผล เปิดเซิร์ฟเวอร์ และติดต่อกับฮาร์ดแวร์ในเวลาเดียวกัน อีกทั้งยังรันในระบบปฏิบัติการ Linux (Ubuntu) ที่ยอดนิยม สามารถศึกษาเพิ่มเติมได้ง่าย และสามารถติดตั้งแพคเกจที่จำเป็นต้องใช้ในการดำเนินโปรเจคได้

ข้อมูลจำเพาะ

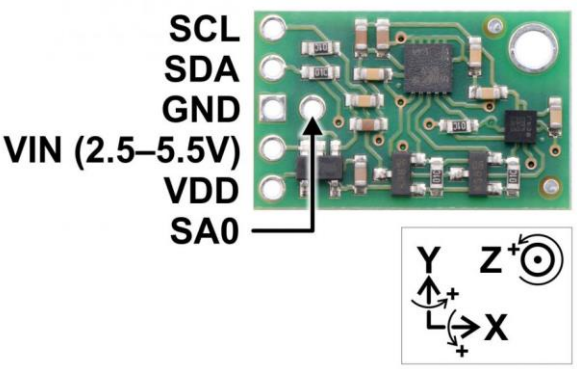
- CPU: Allwinner H2+, Quad-core Cortex-A7
- DDR3 RAM: 256MB/512MB
- Connectivity: 10/100M Ethernet
- Wifi ： XR819
- USB Host: 2.54mm pin x2, exposed in 2.54mm pitch pin header
- MicroSD Slot x 1
- MicroUSB: OTG and power input
- Debug Serial Interface: exposed in 2.54mm pitch pin header
- Audio input/output Interface: exposed in 2.54mm pitch pin header
- GPIO1: 2.54mm spacing 16pin. It includes UART, SPI, I2C, Audio etc
- GPIO2: 2.54mm spacing 16pin. It includes USB,10/100M Ethernet, IO etc
- PCB Dimension: 25.4 x 50mm
- Power Supply: DC 5V/2A
- Temperature measuring range: -40 C to 80 C
- OS/Software: U-boot， Linux Kernel 4.11.2 (mainline) , Ubuntu 16.04.2 LTS (Xenial)



รูป 2.1 แผนภาพโครงสร้าง, ขาเชื่อมต่อของ NanoPI-DUO

2.1.2 เซนเซอร์ (Sensors)

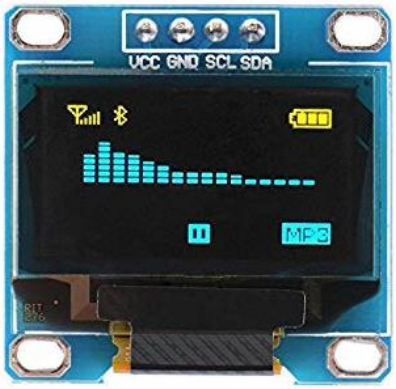
เซนเซอร์ที่ใช้ตรวจจับการเคลื่อนไว้ของมือจะประกอบไปด้วย Flex Sensor เพื่อวัดความงอของแต่ละนิ้วมือ และ IMU (Inertial Measurement Unit: Accelerometer+Gyroscope หรือเซนเซอร์หน่วยวัดความเร่ง) เพื่อใช้ตรวจจับการเคลื่อนไหวและวัดมุมปัจจุบันของมือ



รูป 2.2 Polulu MinIMU9-v5

Polulu MinIMU9-v5 เป็น IMU ตัวที่ใช้ในโครงงานนี้ ซึ่งมีขนาดเล็ก วางบนมือและปลายนิ้วได้ สามารถส่งข้อมูลผ่านบัส I^2C ได้ อีกทั้งยังมี Magnetometer ซึ่งเป็นเซนเซอร์ตรวจจับทิศทางรวมเข้ามาด้วย กลายเป็น 9 แกน ซึ่งสามารถนำไปประยุกต์ใช้ได้ในอนาคตหากจำเป็น

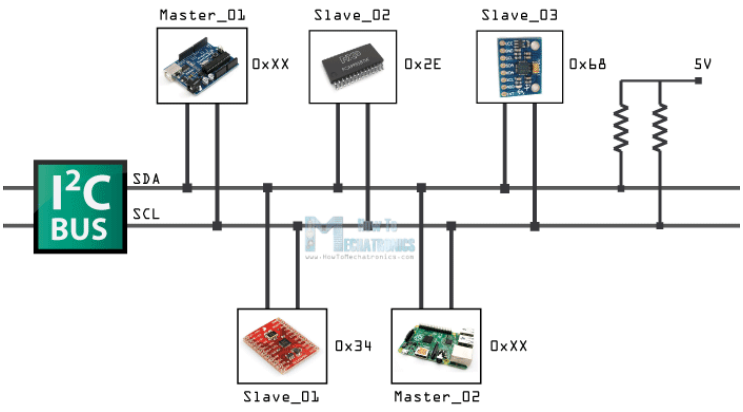
2.1.3 จอแสดงผล (Display)



รูป 2.3 SSD1306

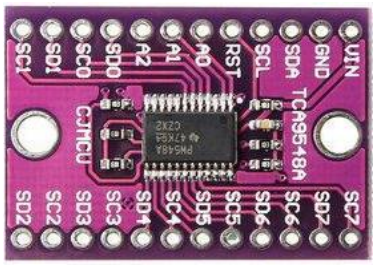
SSD1306 Bi-Color OLED Display เป็นจอแสดงผลสองสี ขนาด 128x64 ซึ่งมีขนาดพอดีสามารถวางไว้บนมือได้ นอกจากนั้นยังใช้การสื่อสารด้วย I^2C เช่นเดียวกันกับเซนเซอร์อื่น ๆ

2.1.4 การติดต่อระหว่างฮาร์ดแวร์



รูป 2.4 ตัวอย่างเครือข่ายการเชื่อมต่อของบัส I^2C

Inter Integrate Circuit (I^2C) เป็นการสื่อสารอนุกรมแบบเข้าสัญญาณ ที่ทั้งสองฝั่งสามารถเป็นได้ทั้งผู้รับและผู้ส่ง เซนเซอร์จะต่อเข้ากับไมโครคอนโทรลเลอร์ผ่านบัส I^2C แล้วไมโครคอนโทรลเลอร์สามารถเขียนและอ่านเซนเซอร์ได้จากการเข้าถึงเลขตำแหน่ง (Address) จำเพาะนั้น ๆ ของเซนเซอร์



รูป 2.5 TCA9548A

โมดูลขยายช่องสัญญาณ I^2C 8 ช่อง (TCA9548A) เนื่องจาก IMU ที่ใช้ในการดำเนินโครงงานอาจมีหลายตัว แต่ทุกตัวที่ใช้นั้นมีเลขตำแหน่งเดียวกัน เพื่อแก้ไขปัญหาที่เลขตำแหน่งซ้ำจึงต้องใช้โมดูลตัวนี้ โดยหลักการการทำงานเหมือน Multiplexer กล่าวคือสามารถเขียนช่องที่ต้องการไปยังตำแหน่งของ TCA9548A เพื่ออ่านค่าจากช่องนั้น ๆ ได้

สำหรับการอ่านข้อมูลจาก I^2C จะใช้ไลบรารี่ SMBus ของ Python ที่เขียนขึ้นมาเพื่อรับส่งข้อมูลด้วย I^2C โดยเฉพาะ

2.2 เทคโนโลยีด้านการประมวลผลท่าทางมือ

การตรวจจับและประมวลผลท่าทางของมือ จะใช้ภาษา Python เวอร์ชั่น 3.6 เนื่องมาจากการที่เป็นภาษายอดนิยมในการพัฒนา Machine Learning ซึ่งส่งผลให้มีเฟรมเวิร์คหรือไลบรารี่ต่าง ๆ ที่สนับสนุนมากมาย

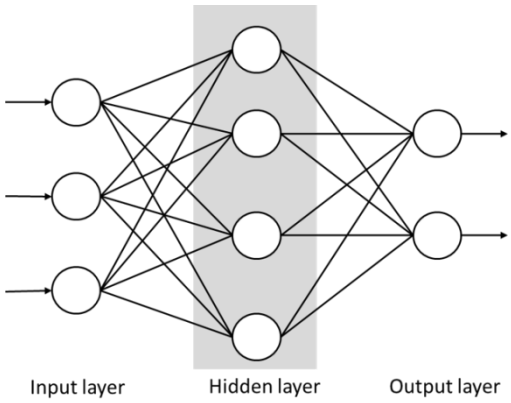
TensorFlow เป็นเฟรมเวิร์คที่ใช้ในการพัฒนาโมเดล Machine Learning เพราะเป็นหนึ่งในเฟรมเวิร์คที่นิยมมากในการพัฒนาโมเดล Machine Learning เนื่องจากมีประสิทธิภาพในเรื่องของความเร็วในการคำนวณอย่างเหมาะสมเมื่อเทียบกับความยากในการเขียน และ TensorFlow ถูกพัฒนาโดยบริษัท Google ทำให้มีการอัพเดทเรื่องของประสิทธิภาพของเฟรมเวิร์คนี้อย่างสม่ำเสมอ

รูป 2.6 โลโก้ TensorFlow

โมเดลสำหรับการพัฒนา Machine Learning เพื่อการทำนายการเคลื่อนไหวของมือ มีทั้งหมด 2 โมเดล ได้แก่

2.2.1 โครงข่ายประสาทเทียม(Artificial Neural Networks - ANN)

เป็นโมเดลทางคณิตศาสตร์ที่จำลองการทำงานของเครือข่ายประสาทในสมองมนุษย์ โดยมีวัตถุประสงค์ที่จะสร้างเครื่องมือซึ่งมีความสามารถในการเรียนรู้การจดจำรูปแบบ (Pattern Recognition) และการสร้างความรู้ใหม่ (Knowledge Extraction) เช่นเดียวกับความสามารถที่มีในสมองมนุษย์



รูป 2.7 สถาปัตยกรรมโครงข่ายประสาทเทียม

จากรูปที่ 2.7 คือโครงสร้างหลักๆของโครงข่ายประสาทเทียม จะมีสามส่วน ได้แก่ ชั้นขาเข้า (Input layer) , ชั้นซ่อน (Hidden layer) และชั้นขาออก (Output layer)

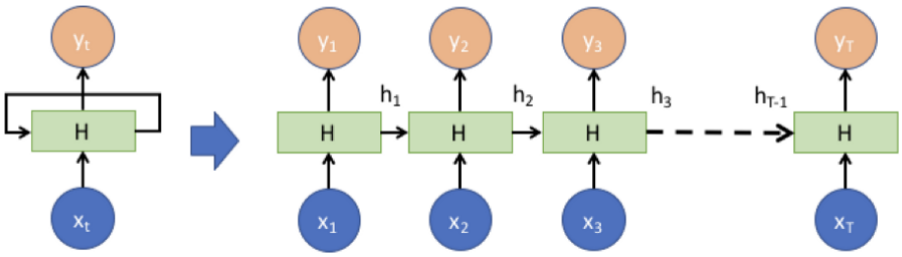
ชั้นขาเข้าจะทำการรับข้อมูลเข้ามาซึ่งอาจมีหลายข้อมูลได้ และชั้นซ่อนจะทำการเอาข้อมูลนั้นมาคูณกับค่าน้ำหนักของแต่ละขาและนำผลลัพธ์ที่ได้จากทุกๆข้อมูลมารวมกันแล้วเอามาเทียบกับค่า threshold ที่กำหนดไว้ ถ้าผลลัพธ์เกินค่า threshold ก็จะทำการส่งผลลัพธ์ออกไป แต่ถ้าค่าน้อยกว่า threshold ก็จะไม่เกิดผลลัพธ์ ซึ่งค่าน้ำหนักจะเป็นค่าที่ไม่แน่นอน แต่สามารถให้คอมพิวเตอร์ปรับค่าได้เองโดยการสอนให้คอมพิวเตอร์รู้จัก pattern ของสิ่งที่เราต้องการให้คอมพิวเตอร์รู้

เรียกว่า "back propagation" ซึ่งจะมีการใช้อัลกอริทึม back-propagation เพื่อใช้ในการปรับปรุงน้ำหนัก หลังจากใส่รูปแบบข้อมูลสำหรับฝึกให้ในแต่ละครั้งแล้ว ค่าที่ได้รับ (output) จากเครือข่ายจะถูกนำไปเปรียบเทียบกับผลลัพธ์ที่คาดหวัง แล้วทำการคำนวณหาค่าความผิดพลาด ซึ่งค่าความผิดพลาดนี้จะถูกส่งกลับเข้าสู่เครือข่ายเพื่อใช้แก้ไขค่าน้ำหนักต่อไป เมื่อปรับค่าน้ำหนักแล้วจะทำให้มีความแม่นยำมากขึ้น

2.2.2 โครงข่ายประสาทเทียมแบบมีหน่วยความจำระยะสั้นและยาว (Long Short-Term Memory)

Long Short-Term Memory (LSTM) เป็นโครงข่ายประสาทเทียมแบบหนึ่งที่ถูกออกแบบมาสำหรับการประมวลผลแบบมีลำดับ(sequence) และเป็นรูปแบบหนึ่งของ Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) เป็นโครงข่ายประสาทเทียมแบบหนึ่งที่เหมาะกับการใช้งานกับข้อมูลที่มีลักษณะเป็นลำดับ (sequence) ซึ่งมันคือการเอาผลลัพธ์ที่ได้จากการคำนวณย้อนกลับมาใช้เป็นข้อมูลขาเข้าอีกครั้ง ดังรูปที่13 ซึ่งมีประโยชน์อย่างมากในข้อมูลที่มีความต่อเนื่อง เช่น time series ข้อมูลเสียง ข้อความ เป็นต้น ดังนั้น 2 ส่วนที่สำคัญของ RNN ก็คือ Hidden state ก่อนหน้า และ input data ณ ตอนนั้น



รูป 2.8 สถาปัตยกรรมโครงข่ายประสาทเทียมแบบ RNN

- H = Hidden layer
- yt = Output จาก RNN ที่เวลา t
- xt = Input data ที่เวลา
- ht = Hidden state ที่เวลา t

ในแต่ละ node ของ RNN จะมีข้อมูลขาเข้าสองอย่างอันได้แก่ input ณ node นั้น ๆ และผลลัพธ์ที่ได้จากการคำนวณใน node ก่อนหน้า ซึ่งทั้งสองข้อมูลจะถูกนำมารวมเข้าด้วยกันและออก

ผลลัพธ์มาเป็นสองทางคือ ผลลัพธ์ที่ออก ณ node นั้น ๆ และออกเพื่อไปเข้าเป็นข้อมูลขาเข้าใน node ถัดไป

ข้อดีของ RNN คือ มันมีการใช้ข้อมูลก่อนหน้าในการทำนายสิ่งที่อาจจะเกิดขึ้นในอนาคต นั่นหมายถึง อะไรที่เคยเกิดขึ้นในอดีตย่อมส่งผลต่อเหตุการณ์ที่จะเกิดขึ้นในอนาคตด้วย แต่แม้ RNN จะมีข้อดีในการทำงานของข้อมูลที่มีความต่อเนื่อง หนึ่งในข้อเสียของ RNN คือ มันสามารถดูย้อนกลับได้แค่เพียงในช่วงระยะเวลาสั้น ๆ เท่านั้น

Long Short-Term Memory ได้มีการเปลี่ยนตัวฟังก์ชันด้านในให้มีความเสถียรและมีประสิทธิภาพมากขึ้นโดยมีหน่วยความจำเข้ามาเกี่ยวข้องที่ไว้คอยจัดการจำสิ่งที่เกิดขึ้นและใช้ในการตัดสินใจในเวลาถัดไป โดยในหน่วยความจำนี้จะต้องถูกปรับไปเรื่อย ๆ การที่มีหน่วยความจำเข้ามาเกี่ยวข้องนั้นทำให้แก้ปัญหาของ RNN ได้คือสามารถดูข้อมูลย้อนกลับได้ระยะที่ยาวมากขึ้น

2.2.2.1 การทำงานของโครงข่ายประสาทเทียมแบบ LSTM

1. Cell state เป็นตัวเก็บ state ของ memory cell ใน LSTM
2. Gate เป็นตัวที่ควบคุมการไหลของข้อมูล ซึ่งก็คือ ค่า analog ที่คอยควบคุมว่าเมื่อไหร่ควรจะ read, write หรือ forget ซึ่งมันก็เหมือนกับประตูที่จะดูว่า เมื่อไหร่ควรเปิดให้ข้อมูลไหลเข้า ไหลออก หรือไหลหายไปเลย (forget)

2.1) Forget gate คือการลบ cell state ออกไปและเตรียมพื้นที่รับข้อมูลใหม่ โดยมี Forget gate เป็นตัวตัดสินใจ ถ้า Forget gate ให้ค่าเป็น 0 ก็ลบ cell state ออกไป แต่ถ้า Forget gate ให้ค่าเป็น 1 ก็จะยังเก็บ cell state เดิมต่อไป ซึ่งข้อมูลที่จำเป็นต้องใช้ในการตัดสินใจนั้นมาจาก input data ที่เข้ามาใหม่ ประกอบกับ hidden state ก่อนหน้า โดยจะใช้ sigmoid function เป็นตัวตัดสิน

2.2) Input gate คือตัวที่ตัดสินว่าจะอนุญาตให้อัพเดท cell state หรือไม่เมื่อมี input data ใหม่เข้ามา การคำนวณนี้ใช้ค่า input data ที่เข้ามา กับ hidden state ก่อนหน้านั้น

2.3) Input modulation gate คือตัวที่ตัดสินใจว่าถ้ามีการอัพเดท cell state จะอัพเดทด้วยค่าอะไร

การอัพเดท cell state คือการนำข้อมูลจาก Forget Gate , Input Gate และ Input modulation Gate มารวมเข้าด้วยกัน ดังนี้

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

กำหนดให้

c_t คือค่าของ cell state

f_t คือค่าของ forget gate

c_{t-1} คือค่าของ cell state ณ เวลาก่อนหน้า

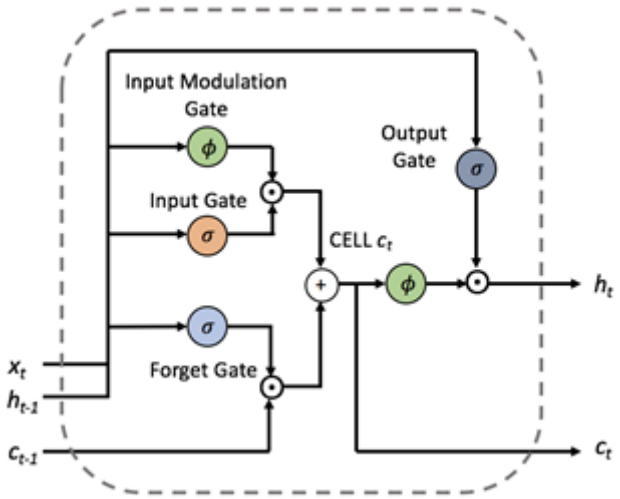
i_t คือค่า input gate

g_t คือค่าที่จะอัพเดท

เริ่มจากส่วนแรกของสมการ ถ้า f_t เป็น 0 ก็จะไม่เอา C_{t-1} มาพิจารณาในการอัพเดท cell state ถ้า f_t เป็น 1 จะยังคงค่า C_{t-1} เอาไว้ประกอบการพิจารณาการอัพเดท ส่วนหลังของสมการส่วนนี้จะเป็นส่วนของการอัพเดท cell state จากข้อมูลใหม่ ซึ่งตอนนี้มีค่าที่จะอัพเดทจาก g_t แล้ว และจะใช้ i_t มาเป็นตัวตัดสินในการอัพเดท ถ้าเป็น 1 ก็อัพเดทได้เลย แต่ถ้าเป็น 0 ก็จะไม่มีการอัพเดท จากค่าทั้งหมดทำให้ได้ค่า c_t ตัวใหม่

2.4) Output gate คือสิ่งที่เราต้องผลิตออกไปก็คือ hidden state ณ เวลาที่ t หรือ ht ซึ่งเมื่อตอนที่เวลา t+1 ตัว LSTM จะเอาค่า ht นี้ไปคำนวณด้วย ซึ่งการส่งค่า ht ออกไปนั้นจะมีตัว Output Gateเป็นตัวตัดสินใจ และจะนำค่า output ไปเป็นค่า ht (Input) สำหรับ sequence ถัดไป

จะเห็นได้ว่า LSTM จะเหมาะกับข้อมูลที่เข้ามาเป็นลำดับและสามารถนำข้อมูลเก่ามาคำนวณซึ่งอาจจะมีผลต่อในอนาคตได้ โมเดลนี้จึงเหมาะที่จะนำมาทำนายคุณภาพอากาศในอนาคตเนื่องจากข้อมูลที่เข้ามาเป็นข้อมูลที่เป็นลำดับและมีความต่อเนื่อง



รูป 2.9 สถาปัตยกรรมโครงข่ายประสาทเทียมแบบ LSTM

2.3 เทคโนโลยีด้านฐานข้อมูล

รูป 2.10 โลโก้ MongoDB

MongoDB เป็นระบบจัดการฐานข้อมูลแบบไม่เป็นตาราง (Non-relational Database) ที่ยอดนิยม ซึ่งมีประโยชน์อย่างมากในการดำเนินโครงงาน เนื่องจากมีข้อมูลบางชนิดที่ไม่ทราบขนาด การใช้ฐานข้อมูลแบบนี้และเก็บข้อมูลเป็นวัตถุแทนจะทำให้ดำเนินการเก็บและเรียกค้นได้ง่ายขึ้น

2.4 เทคโนโลยีด้านแอพพลิเคชั่น

ในการพัฒนาแอพพลิเคชั่นโต้ตอบกับผู้ใช้ ผู้พัฒนาเลือกที่จะทำเป็นเว็บแอพพลิเคชั่น เนื่องจากเมื่อผู้ใช้เชื่อมต่อกับถุงมือ ผู้ใช้ก็จะสามารถเข้าถึงได้ทันทีผ่านอุปกรณ์ทั่วไป เช่น คอมพิวเตอร์, โทรศัพท์ และแท็บเล็ต โดยไม่ต้องดาวน์โหลดแอพพลิเคชั่นใด ๆ ล่วงหน้า

รูป 2.11 โลโก้ VueJS

VueJS เป็นเฟรมเวิร์คที่ใช้ในการพัฒนาเว็บแอพพลิเคชั่นฝั่งผู้ใช้ เขียนโดยใช้ภาษา Javascript โดยเป็นเฟรมเวิร์คสามารถพัฒนาเว็บแอพพลิเคชั่นที่สามารถเปลี่ยนแปลงข้อมูลได้แบบเรียลไทม์ จึงเหมาะสมสำหรับการพัฒนาที่ต้องมีการป้อนและอัพเดทข้อมูลจากเซนเซอร์ตลอดเวลา

รูป 2.12 โลโก้ Flask

Flask เป็นเฟรมเวิร์คใช้ในการพัฒนาเว็บแอพพลิเคชั่นฝั่งเซิร์ฟเวอร์ เขียนโดยใช้ภาษา Python สามารถติดต่อฐานข้อมูล MongoDB ได้ง่าย และด้วยเนื่องจากการประมวลผลข้อมูลทำโดยใช้ภาษา Python จึงควรใช้เฟรมเวิร์คสำหรับพัฒนา Server เป็นภาษาเดียวกันด้วย ถึงจะสามารถทำงานร่วมกันได้

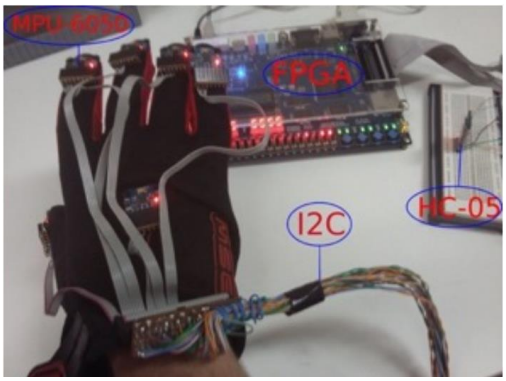
Socket.IO เป็นโมดูลที่ทำให้เว็บแอพพลิเคชั่นฝั่งผู้ใช้สามารถเชื่อมต่อกับเซิร์ฟเวอร์แบบเรียลไทม์ได้ เมื่อมีการเชื่อมต่อตั้งแต่เริ่มเข้าเว็บแอพพลิเคชั่น ทั้งสองฝั่งสามารถส่งข้อมูลหากันได้ตลอดเวลา และใช้เวลาน้อย เพราะไม่จำเป็นจะต้องเปิดการเชื่อมต่อทุกครั้งที่ส่งข้อมูลดังเช่นการส่งข้อมูลแบบ HTTP Request ที่นิยมทั่วไป

Eventlet เป็นไลบรารี่สำหรับช่วยในการทำงานแบบ Multithread ช่วยให้สามารถทำการเปิดเซิร์ฟเวอร์โดยใช้ Flask และรอข้อความจาก Socket.IO พร้อมกับการประมวลผลข้อมูลด้วย Machine Learning ไปพร้อม ๆ กันได้

2.5 งานวิจัยที่เกี่ยวข้อง

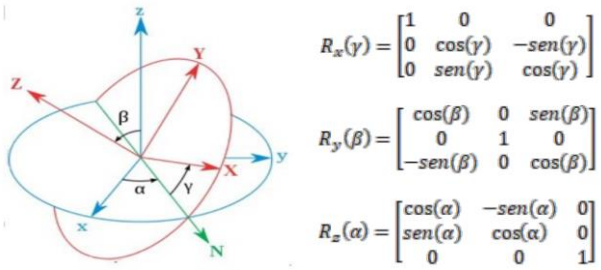
2.5.1 A Gesture Detection Glove For Human-computer Interaction

งานวิจัยนี้มีเป้าหมายเพื่อประดิษฐ์ถุงมือเพื่อตรวจจับท่าทางต่าง ๆ ในการควบคุมเมาส์และคีย์บอร์ดผ่านการเชื่อมต่อแบบ Bluetooth โดยใช้ MCU เป็น PAMPIUM ที่ประกอบจาก FPGA พัฒนาโดยมหาวิทยาลัย Pampa และใช้เซนเซอร์ IMU ที่กลางมือและปลายนิ้วทั้ง 5 รวม 6 ตัว



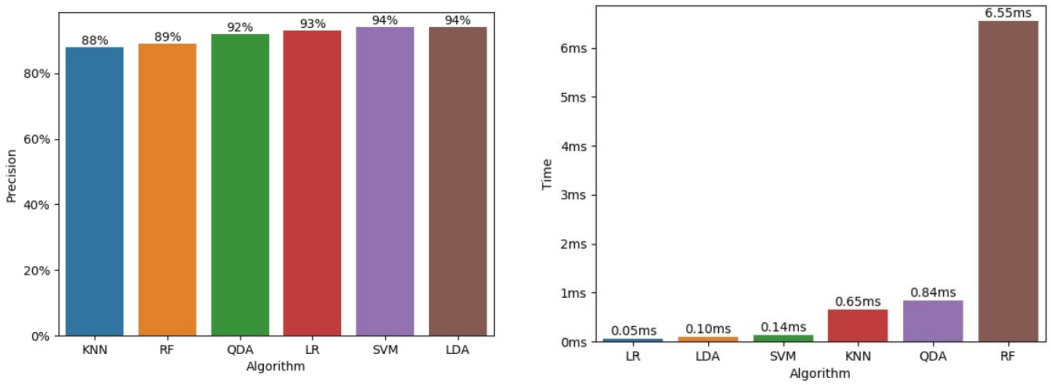
รูป 2.13 ถุงมือของงานวิจัย A Gesture Detection Glove For Human-computer Interaction

ค่าที่ได้จากเซนเซอร์ IMU จะ ส่วนของ Accelerometer จะถูกปรับให้อยู่ในค่าระหว่าง -10 ถึง +10 และ Gyroscope จะถูกปรับให้อยู่ในค่าระหว่าง -π ถึง +π จากนั้นจะถูกนำมาแปลงเป็น Rotation Matrix



รูป 2.14 Rotation Matrix

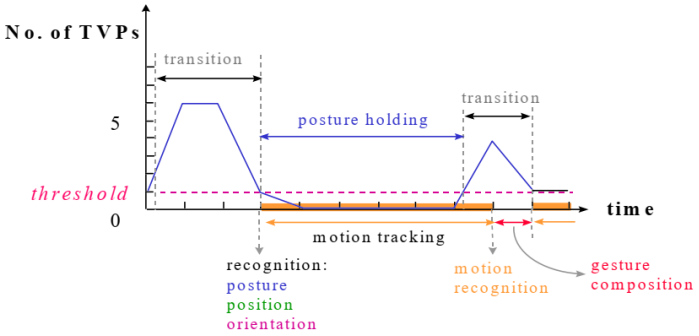
การตรวจจับท่าทางมือจะเป็นการพัฒนาและใช้โมเดล Machine Learning หลากหลายแบบ ได้แก่ KNN, RF, QDA, LR, SVM และ LDA เพื่อนำมาเปรียบเทียบความแม่นยำและเวลาที่ใช้



รูป 2.15 แผนภูมิเปรียบเทียบความแม่นยำและเวลาที่ใช้ระหว่างโมเดล Machine Learning แต่ละตัวของงานวิจัย A Gesture Detection Glove For Human-computer Interaction

2.5.2 A real-time continuous gesture recognition system for sign language

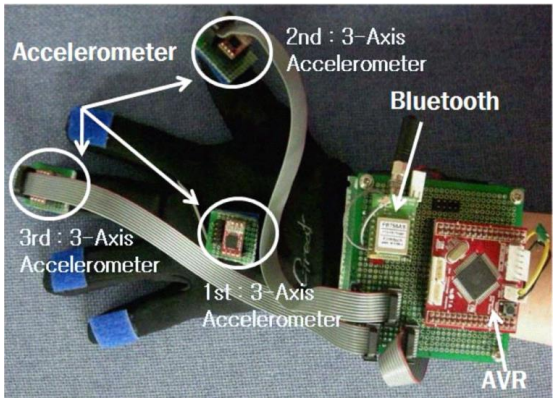
งานวิจัยนี้มีเป้าหมายเพื่อประดิษฐ์ถุงมือเพื่อตรวจจับท่าทางของมือ 40 ท่า เพื่อแปลเป็นศัพท์ตามภาษามืออเมริกัน โดยการหาคุณลักษณะจากชุดข้อมูลเซนเซอร์ตามเวลา นำเข้าไปฝึกในโมเดลโครงข่ายประสาทเทียมแบบ Markov Model



รูป 2.16 แผนภูมิเวลาในการวิเคราะห์ท่าของงานวิจัย A real-time continuous gesture recognition system for sign language

2.5.3 3-D hand motion tracking and gesture recognition using a data glove

งานวิจัยนี้มีเป้าหมายเพื่อประดิษฐ์ถุงมือเพื่อตรวจจับการเคลื่อนไหวของมือและท่าทางต่าง ๆ โดยใช้ MCU เป็น ATmega128 (AVR) ใช้เซนเซอร์ IMU ที่กลางมือ, ปลายนิ้วโป้ง และปลายนิ้วกลาง รวม 3 ตัว และติดต่อกับคอมพิวเตอร์ผ่านสัญญาณ Bluetooth



รูป 2.17 ถุงมือจากงานวิจัย 3-D hand motion tracking and gesture recognition using a data glove

การตรวจจับการเคลื่อนไหว จะใช้การเทียบค่าจากเซนเซอร์กับมุมต่าง ๆ ของมือ ส่วนการตรวจจับท่าทาง จะใช้การตรวจสอบเงื่อนไขว่าหากค่าตรงกับเงื่อนไขของท่าใด ก็จะตรวจจับได้ตรงกับท่านั้น ๆ การทดลองของงานวิจัยพบว่าหลังจากการทดสอบท่าละ 50 ครั้ง สามารถแยกท่าเป่ายิ้งฉุบ 3 ท่า ได้แก่ ค้อน กระดาษ กรรไกร ออกจากกันได้ร้อยละ 100

บทที่ 3
การออกแบบและพัฒนา

3.1 การออกแบบถุงมือ

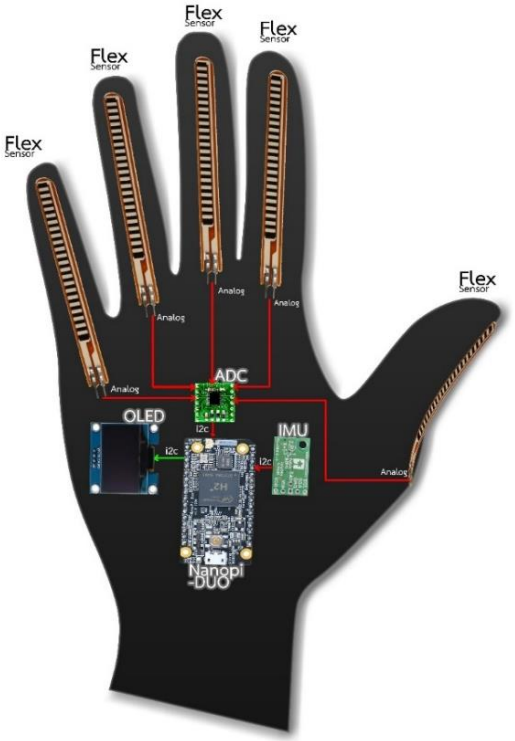
สำหรับถุงมือ ผู้พัฒนาได้ออกแบบไว้ 2 แบบ คือแบบที่ใช้เซนเซอร์วัดความงอสำหรับนิ้ว และเซนเซอร์ IMU สำหรับวัดมุมของมือ และอีกแบบคือแบบที่ใช้ IMU วัดทั้งความงอและมุมของมือ ถุงมือทั้ง 2 แบบ จะมีรูปแบบข้อมูลที่ได้รับแตกต่างกัน และส่งผลให้การประมวลผลในโมเดล Machine Learning ต่างกันด้วย

3.1.1 ถุงมือแบบที่ 1:

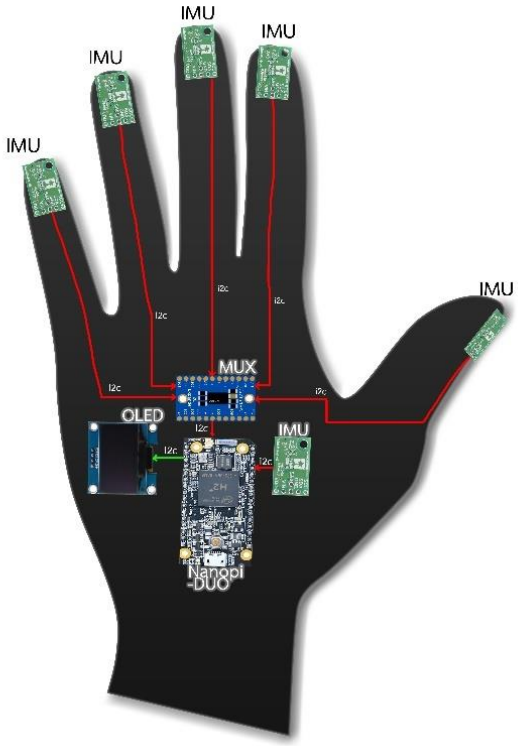
เป้าหมายของถุงมือรูปแบบนี้ คือเพื่อความแม่นยำของการงอนิ้ว 1 แกนที่เป็นแกนหลัก นั่นคือแกนในการกำมือ และเพื่อความง่ายในการพัฒนา เซนเซอร์ที่ใช้จะมีทั้งหมด 6 ตัว ประกอบไปด้วยเซนเซอร์ตรวจจับการงอ (Flex Sensor) ของนิ้ว 5 ตัว วางตามนิ้วแต่ละนิ้ว และเซนเซอร์ IMU ตรวจสอบการเคลื่อนไหวและการวางมือ วางไว้กลางหลังมือ 1 ตัว

3.1.2 ถุงมือแบบที่ 2:

เป้าหมายของถุงมือรูปแบบนี้ คือเพื่อความแม่นยำของตำแหน่งนิ้วในทุกแกน เซนเซอร์ที่ใช้จะมีทั้งหมด 6 ตัว ประกอบไปด้วยเซนเซอร์ IMU ของนิ้ว 5 ตัว วางตามปลายนิ้วบริเวณเล็บของแต่ละนิ้ว และเซนเซอร์ IMU ตรวจสอบการเคลื่อนไหวและการวางมือ วางไว้กลางหลังมือ 1 ตัว



รูป 3.1 โครงสร้างถุงมือแบบที่ 1



รูป 3.2 โครงสร้างถุงมือแบบที่ 2

3.2 การออกแบบระบบโดยรวม

เพื่อให้เห็นได้ชัดเจน ระบบสามารถแบ่งออกได้เป็น 3 ส่วน คือฮาร์ดแวร์, เซิร์ฟเวอร์ และเว็บแอพพลิเคชั่น

3.2.1 ฮาร์ดแวร์ (Hardware)

คือส่วนของถุงมือทั้งหมด รวมถึงเซนเซอร์และ MCU มีหน้าที่เก็บตัวอย่างข้อมูล (Sample) จากการทำท่าทางต่าง ๆ ของผู้ใช้ และแสดงผลข้อมูลต่าง ๆ ตามที่ได้คำสั่งจากเซิร์ฟเวอร์

3.2.2 เซิร์ฟเวอร์ (Server)

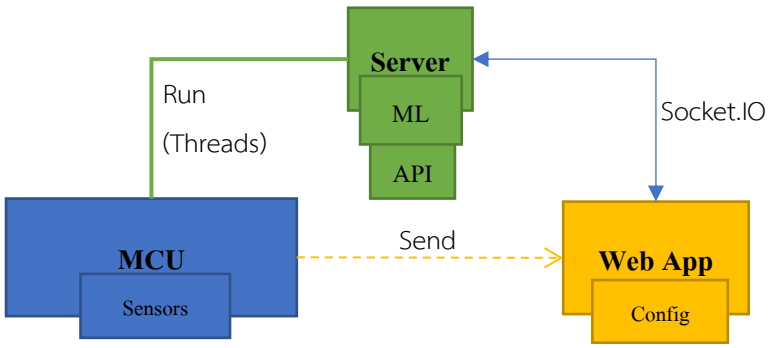
คือส่วนซอฟต์แวร์ที่รันบน MCU ตอดเวลา เขียนด้วยภาษา Python และมีหน้าที่ดังต่อไปนี้

- อ่านข้อมูลจากฮาร์ดแวร์ ทำการคำนวณเพื่อแปลงข้อมูลหรือเพื่อความแม่นยำหากจำเป็น
- ตรวจสอบข้อมูลจากฮาร์ดแวร์เพื่อหาช่วงของชุดข้อมูลที่ต้องนำไปคำนวณในโมเดล Machine Learning
- นำชุดข้อมูลไปคำนวณในโมเดล Machine Learning และรอรับคำตอบ
- ส่งเว็บแอพพลิเคชั่นให้ผู้ใช้หากมีการเรียกใช้งาน
- คอยรับและตอบคำร้องขอข้อมูลผ่าน HTTP (HTTP Request) จากเว็บแอพพลิเคชั่นฝั่งผู้ใช้
- คอยรับและส่งข้อมูลผ่าน WebSocket จากเว็บแอพพลิเคชั่นฝั่งผู้ใช้

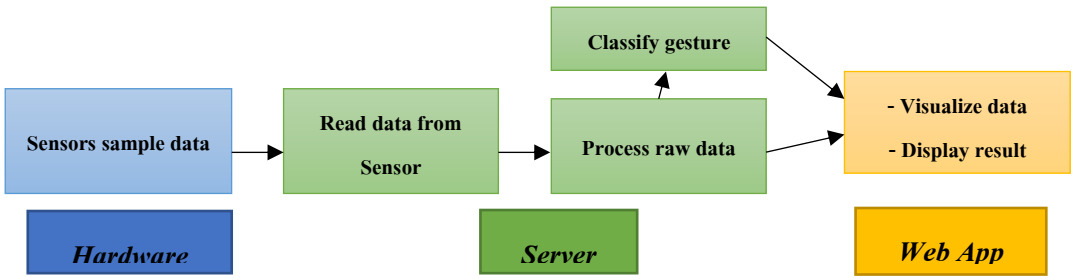
3.2.3 เว็บแอพพลิเคชั่น (Web Application)

คือส่วนที่จะทำงานก็ต่อเมื่อผู้ใช้ส่งคำขอ HTTP (HTTP Request) ให้แก่เซิร์ฟเวอร์ แล้วเซิร์ฟเวอร์ก็จะส่งคำตอบรับ (HTTP Response) เป็นเว็บแอพพลิเคชั่น และทำงานอยู่บนบราวเซอร์ของผู้ใช้เอง ส่วนนี้มีหน้าที่เบื้องต้นดังนี้

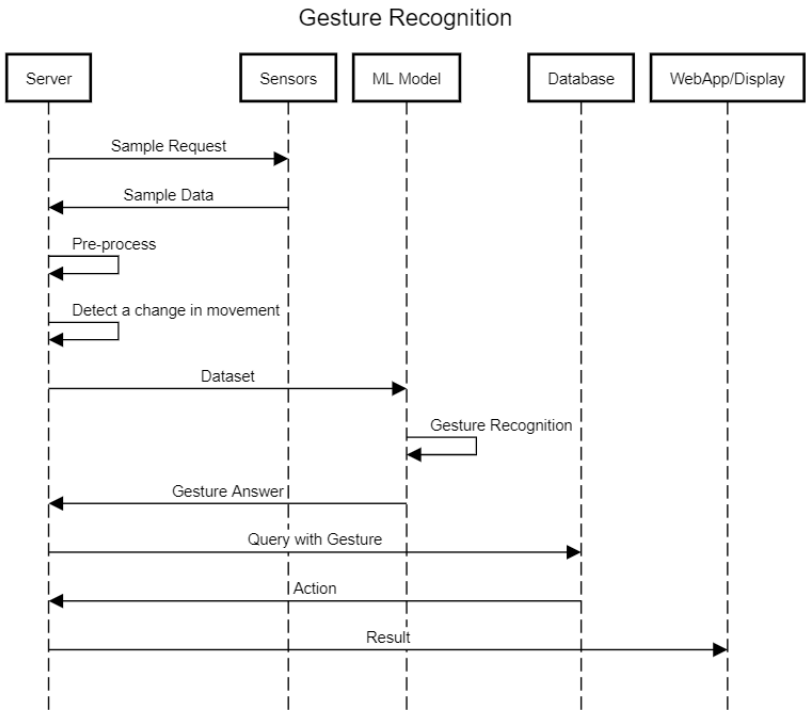
- เชื่อมต่อ รับ และส่งข้อมูลผ่าน WebSocket จากเซิร์ฟเวอร์
- แสดงผลข้อมูล/ผลการตรวจจับท่าทางที่ส่งมาจากเซิร์ฟเวอร์
- เป็นส่วนที่ติดต่อกับผู้ใช้ (UI) สำหรับการตั้งค่าต่าง ๆ ที่เกี่ยวกับระบบโยยรวม



รูป 3.3 โครงสร้างระบบ



รูป 3.4 ขั้นตอนการทำงานแบ่งตามส่วนของการตรวจจับท่าทาง



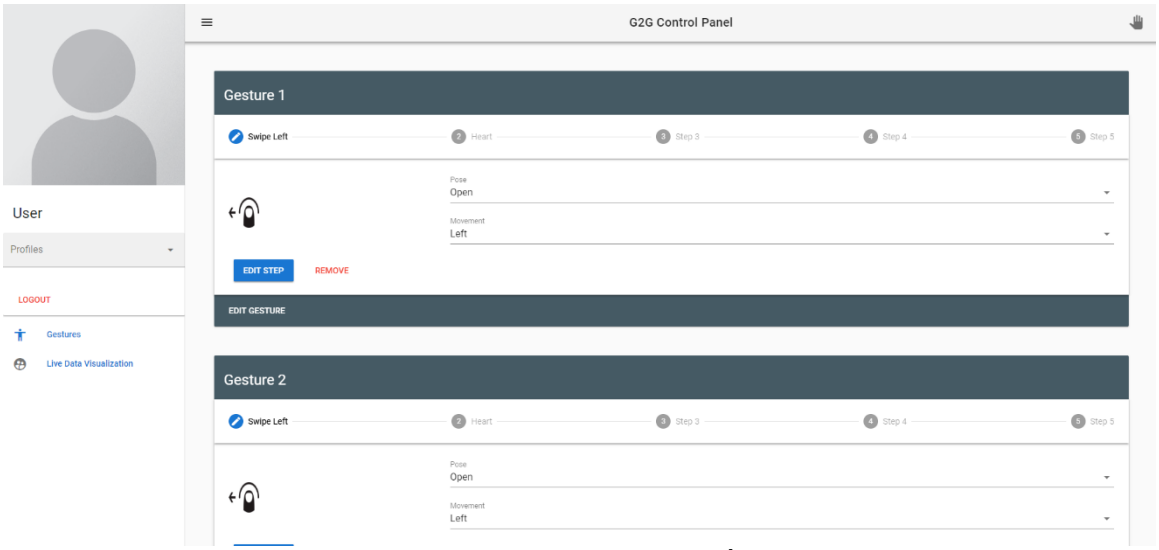
รูป 3.5 แผนผังแสดงลำดับการทำงานของการตรวจจับท่าทาง

3.3 การออกแบบเว็บแอพพลิเคชั่น

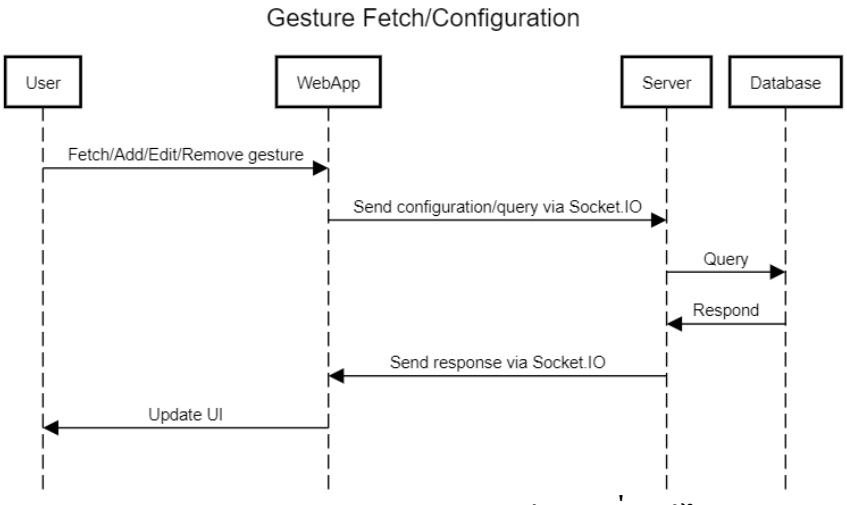
เว็บแอพพลิเคชั่นจะแบ่งออกเป็นส่วนหลัก ๆ 2 ส่วน คือส่วนของผู้ใช้ และส่วนของนักพัฒนา

3.3.1 ส่วนของผู้ใช้

มีจุดหมายเพื่อให้ผู้ใช้งานถุงมือตั้งค่าท่าทางต่าง ๆ ว่าต้องการทำท่าแบบใดและต้องการให้แสดงผลหรือเกิดสิ่งใดขึ้น โดยจะมีการแสดงรายการท่าทางและขั้นตอนในการทำท่าทางนั้น ๆ เรียงกัน ผู้ใช้สามารถเพิ่ม แก้ไข ลบขั้นตอนต่าง ๆ ได้



รูป 3.5 ต้นแบบหน้าตาเว็บแอพพลิเคชั่นส่วนของผู้ใช้

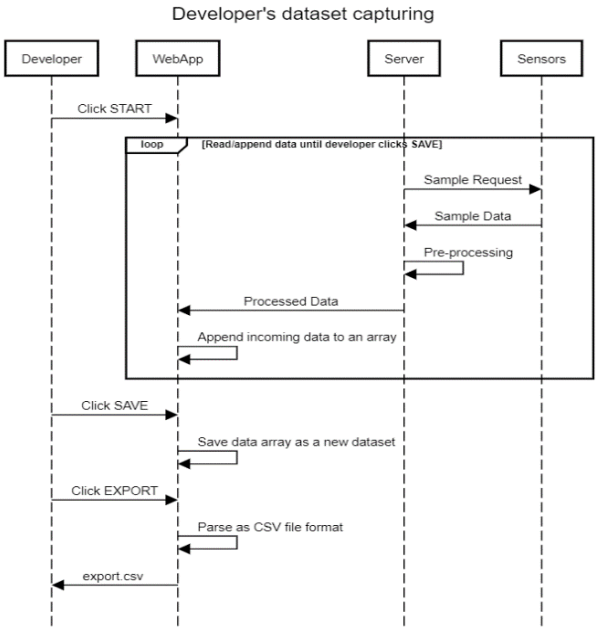


รูป 3.6 แผนผังแสดงลำดับการทำงานของการเรียกดู/เพิ่ม/แก้ไข/ลบ ท่าทางของผู้ใช้

3.3.1 ส่วนของนักพัฒนา

มีจุดหมายเพื่อแสดงผลข้อมูลจากเซนเซอร์ แสดงผลข้อมูลจากฐานข้อมูล ทดสอบส่วนต่าง ๆ ของระบบ บันทึกชุดข้อมูลสำหรับสร้างโมเดล Machine Learning และรวมถึงสิ่งที่มีประโยชน์ต่าง ๆ ที่ใช้ในกระบวนการพัฒนา

รูป 3.7 ต้นแบบหน้าตาเว็บแอพพลิเคชั่นส่วนของนักพัฒนา



รูป 3.8 แผนผังแสดงลำดับการทำงานของการบันทึกและส่งออกชุดข้อมูลของนักพัฒนา

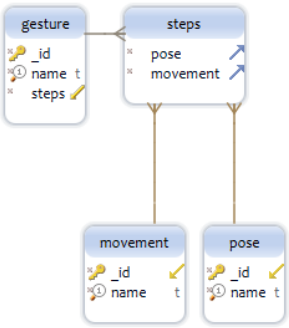
3.4 การออกแบบฐานข้อมูล

สำหรับการเก็บข้อมูลท่า จะใช้ Collection ทั้งหมด 4 Collection ประกอบไปด้วย

Gesture: ชุดท่าทาง 1 ชุด ประกอบไปด้วย steps หลายท่าที่ทำต่อเนื่องกัน

Steps: ขั้นตอน 1 ขั้นตอนของการทำท่า ประกอบไปด้วย pose และ movement อย่างละแบบ

Pose: การวางมือ และ Movement: การขยับมือ



รูป 3.9 แผนภาพฐานข้อมูล

3.5 การออกแบบชุดข้อมูล

ชุดข้อมูลที่จะส่งไปใช้ในการฝึกโมเดลและการตรวจจับท่าทาง จะเป็นชุดของค่าที่จับได้จากเซนเซอร์ติดต่อกัน 20 ค่า ที่แสดงการเปลี่ยนแปลงของค่าเซนเซอร์ติดต่อกันภายในเวลา 20 วินาที จึงทำให้ชุดข้อมูลหนึ่งจะมี 20 ชุดย่อย แต่โครงสร้างของชุดย่อยจะแตกต่างกันไปตามรูปแบบของถุงมือทั้งสองแบบดังนี้

3.1.1 การออกแบบชุดข้อมูลของถุงมือแบบที่ 1:

ชุดย่อยจะประกอบไปด้วยค่ามุมจากเซนเซอร์ 3 แกน และค่าจากเซนเซอร์วัดความงอตามนิ้ว 5 นิ้ว รวมเป็น 8 ค่า ชุดย่อยสามารถแสดงเป็นอาร์เรย์ได้ดังนี้

[มือ_x, มือ_y, มือ_z, โป้ง, ชี้, กลาง, นาง, ก้อย]

3.1.2 การออกแบบชุดข้อมูลของถุงมือแบบที่ 2:

ชุดย่อยจะประกอบไปด้วยค่ามุมจากเซนเซอร์ตัวละ 3 แกน รวมเป็น 18 ค่า ชุดย่อยสามารถแสดงเป็นอาร์เรย์ได้ดังนี้

[มือ_x, มือ_y, มือ_z, โป้ง_x, โป้ง_y, โป้ง_z, ชี้_x, ชี้_y, ชี้_z, กลาง_x, กลาง_y, กลาง_z, นาง_x, นาง_y, นาง_z, ก้อย_x, ก้อย_y, ก้อย_z]

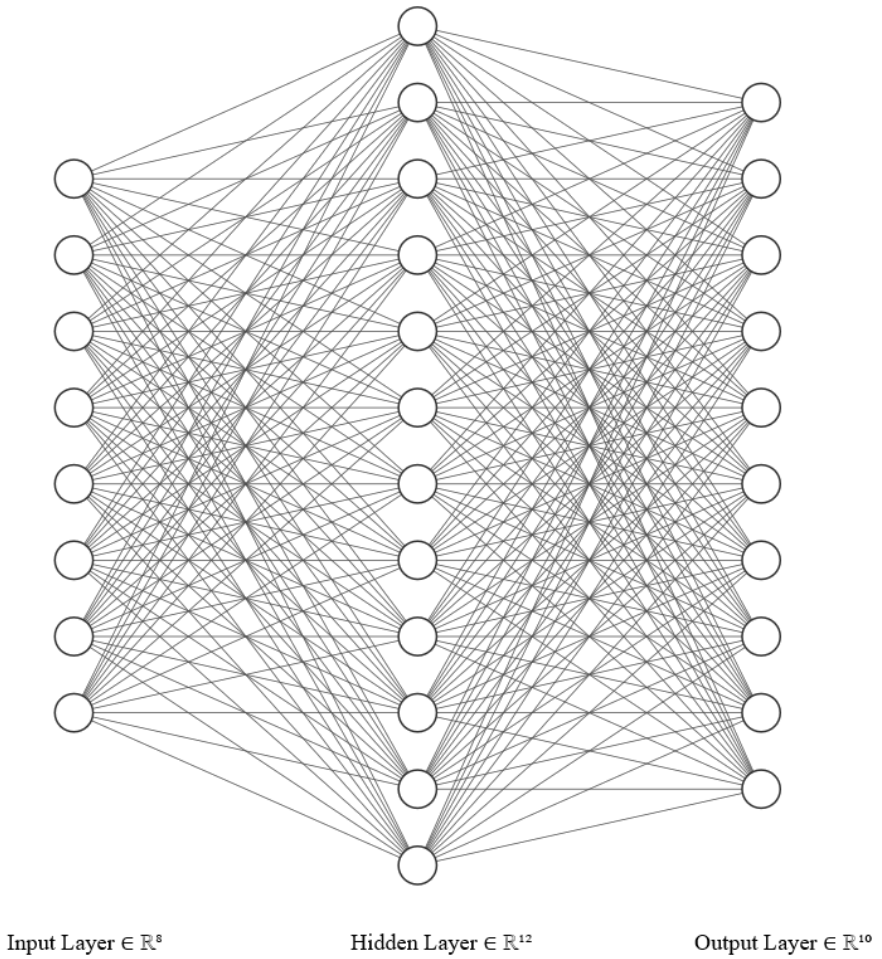
ดังนั้นหากนำชุดย่อยมาต่อกัน 20 ตัว จะกลายเป็นชุดข้อมูล 1 ชุด และในการฝึกโมเดล จะใช้ชุดข้อมูลทั้งหมดอย่างน้อย 50 ตัว

3.6 การออกแบบโมเดล Machine Learning

โมเดลที่ใช้ในการฝึกท่าทางที่มีข้อมูลของเซนเซอร์ในแต่ละนิ้ว เลือกเป็นโครงค่ายประสาทเทียม (Neural Network) เพราะมีชุดข้อมูลเยอะในการฝึกท่าทาง โครงข่ายประสาทเทียมช่วยทำให้ไม่ต้องหาคุณลักษณะ (Feature) ให้กับข้อมูล และใช้การเรียนแบบมีการสอน (Supervised Learning) ก็คือการเรียนที่มีการตรวจคำตอบเพื่อให้โครงข่ายประสาทเทียมปรับตัวเอง เพื่อให้ได้คำตอบที่ดีขึ้น โดยโมเดลสำหรับข้อมูลที่ใช้ในการฝึกจะทดลองทั้งหมด 3 โมเดลดังนี้

3.6.1 การออกแบบโมเดลของถุงมือแบบที่ 1 สำหรับท่านิ่ง:

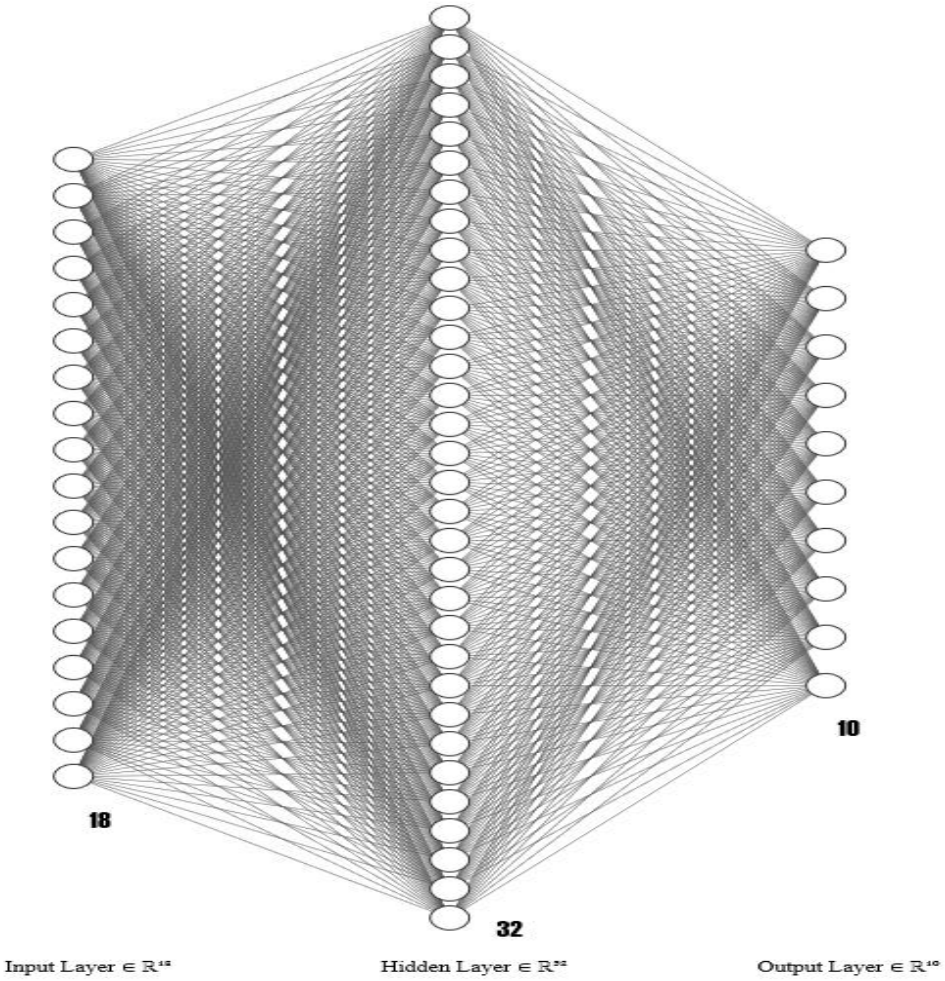
เป็นโมเดลโครงข่ายประสาทเทียมธรรมดา (Neural Network) จะใช้ Library ของ Scikit-learn ในการสร้างโมเดล มีคุณลักษณะ (Feature) ทั้งหมด 8 ลักษณะ ประกอบด้วยเซนเซอร์วัดความงอตามนิ้วทั้งหมด 5 ลักษณะ เซนเซอร์วัดมุมข้อมือ 3 ลักษณะ เพื่อให้ได้ผลลัพธ์ (Output) เป็นท่าทางของมือรูปแบบนิ่ง ก่อนที่จะนำคุณลักษณะทั้งหมดไปฝึกจำเป็นต้องแปลง (Transform) ก่อนเพื่อให้การทำนายผลลัพธ์แม่นยำยิ่งขึ้น โดยโครงสร้างของโครงข่ายประสาทเทียมประกอบด้วย ชั้นขาเข้า (Input layer) คือลักษณะทั้งหมด 8 ชั้น ชั้นซ่อน (Hidden layer) ใช้เป็น 12 ชั้น



รูป 3.10 สถาปัตยกรรมโครงข่ายประสาทเทียมโมเดลที่ 1

3.6.2 การออกแบบโมเดลของถุงมือแบบที่ 2 สำหรับท่านิ่ง:

เป็นโมเดลโครงข่ายประสาทเทียมธรรมดา จะใช้ Library ของ Keras ซึ่งมี Tensorflow เป็น Back-end ในการสร้างโมเดล มีคุณลักษณะ (Feature) ทั้งหมด 18 ลักษณะ ประกอบด้วย เซนเซอร์วัดมุมของแต่ละนิ้วทั้งหมด 15 ลักษณะ เซนเซอร์วัดมุมข้อมือ 3 ลักษณะ เพื่อให้ได้ผลลัพธ์เป็นท่าทางของมือรูปแบบนิ่ง โครงสร้างของประสาทเทียมประกอบด้วย ชั้นขาเข้าคือลักษณะทั้งหมด 18 ชั้น และชั้นซ่อนใช้เป็น 32 ชั้น ส่วนชั้นขาออก (Output layer) ใช้ตามจำนวนของท่าทั้งหมดที่ฝึกเป็น 10 ชั้น

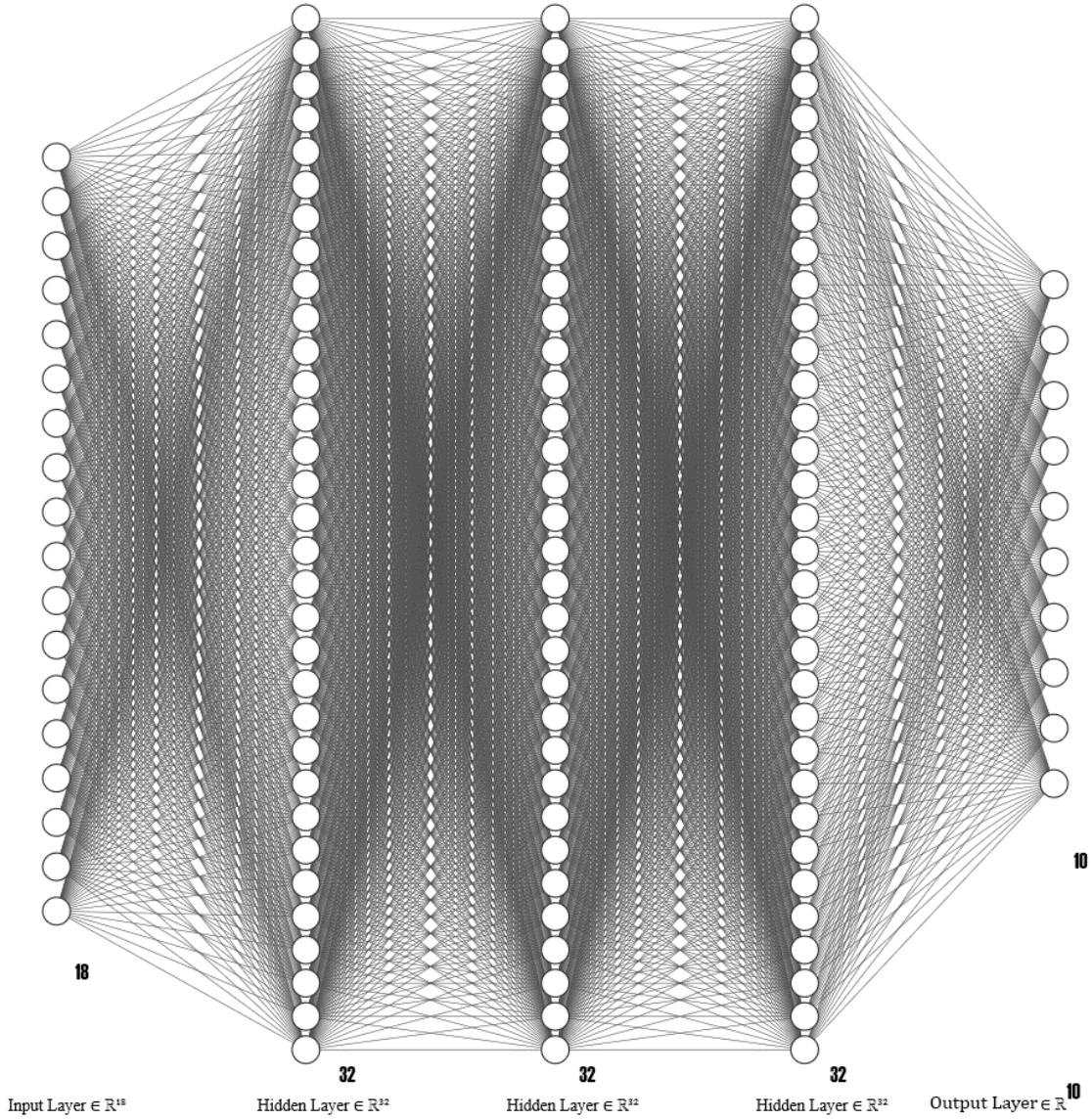


รูป 3.11 สถาปัตยกรรมโครงข่ายประสาทเทียมโมเดลที่ 2

3.6.3 การออกแบบโมเดลของถุงมือแบบที่ 2 สำหรับท่านิ่งและท่าเคลื่อนไหว:

เป็นโมเดลโครงข่ายประสาทเทียมที่ออกแบบมาสำหรับประมวลผลลำดับ (LSTM) จะใช้ Library ของ Keras ในการสร้างโมเดล มีคุณลักษณะทั้งหมด 18 ลักษณะ ประกอบด้วยเซนเซอร์วัด

มุมของแต่ละนิ้วทั้งหมด 15 ลักษณะ เซนเซอร์วัดมุมข้อมือ 3 ลักษณะ โดยมีลำดับของท่าทางทั้งหมด 20 ครั้ง (Timestep) เพื่อให้ได้ผลลัพธ์เป็นท่าทางของมือทั้งท่านิ่งและเคลื่อนไหว โครงสร้างของประสาทเทียมประกอบด้วย ชั้นขาเข้าเป็นอาเรย์ลักษณะ (20,18) 20 ก็คือจำนวนครั้งของลำดับ ส่วน 18 ก็คือคุณลักษณะทั้งหมด ชั้นซ่อนใช้เป็น 32 ชั้น ทั้งหมด 3 เลเยอร์ และชั้นขาออกใช้เป็นจำนวนท่าทั้งหมดในการฝึก มีทั้งหมด 10 ชั้น



รูป 3.12 สถาปัตยกรรมโครงข่ายประสาทเทียมโมเดลที่ 3

บทที่ 4
การทดลอง

การทดลองแบ่งได้เป็น 3 ส่วนหลัก ๆ คือ

4.1 การทดลองเกี่ยวกับฮาร์ดแวร์

คือการทดลองเกี่ยวกับถุงมือที่ได้ออกแบบมาทั้งสองแบบ ว่าสามารถจับค่าได้แม่นยำและมีปัญหาในการจับค่าหรือไม่

4.1.1 การทดลองใช้ถุงมือแบบที่ 1

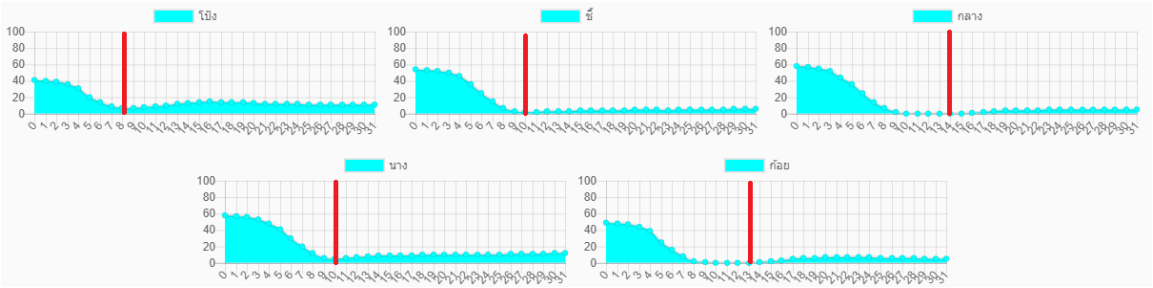
4.1.1.1 วิธีการทดลอง

ประดิษฐ์ถุงมือแบบที่ 1 ขึ้นตามที่ออกแบบไว้ จากนั้นลองอ่านข้อมูลที่ได้จากเซนเซอร์ทุกตัว (Flex Sensor 5 ตัว และ IMU 1 ตัว) สำหรับ Flex Sensor จะมีค่าตั้งแต่ 0-1023 และ IMU จะอ่านเพียงค่าดิบของ Accelerometer 3 แกนและ Gyroscope 3 แกน ทั้งสองมีค่าตั้งแต่ -2000 ถึง 2000 จากนั้นลองทำท่านิ่งเบื้องต้น ได้แก่ท่ากำมือ แบมือ ชูนิ้วแต่ละนิ้ว ชูนิ้วชี้กับนิ้วกลางพร้อมกัน และไขว้นิ้ว

4.1.1.2 ผลการทดลอง

ถุงมือไม่มีปัญหาด้านการจับค่า สามารถทำการจับค่าได้ตามปกติ ยกเว้นในบางกรณีที่จับค่าไม่ได้หรือเพี้ยน ดังนี้

1.) กรณีแบมือจนสุด จะทำให้ Flex Sensor เกิดการงอ ถุงมือจะจับค่าได้เท่ากับการงอนิ้ว จากรูป 4.1 สามารถสังเกตได้ว่า หลังจากเส้นสีแดง ค่ามีแนวโน้มเพิ่มขึ้นเล็กน้อย เกิดจากการงอของ Flex Sensor เมื่อแบมือมากเกินไป ในผู้ใช้บางรายอาจจะแบมือได้มากกว่านี้ ซึ่งอาจจะทำให้ไม่สามารถแยกท่าบางท่าออกจากกันได้



รูป 4.1 แผนภาพแสดงชุดข้อมูลของท่าแบมือ (ถุงมือแบบที่ 1)

2.) กรณีที่มีการไขว้นิ้ว ค่าจะอยู่นอกแกนที่ Flex Sensor สามารถจับได้ ตัวอย่างเช่นการไขว้นิ้วชี้กับนิ้วกลาง ค่าจะได้ใกล้เคียงกับการกางนิ้วทั้งสองปกติ

3.) กรณีที่งอนิ้วโป้งไปหลายทิศทาง ค่าจะอยู่นอกแกนที่ Flex Sensor สามารถจับได้เช่นกัน ตัวอย่างเช่น มุมระหว่างนิ้วโป้งกับนิ้วชี้ การกางเป็นมุมฉากกับการแนบชิดนิ้วทั้งสองจะได้ค่าเหมือนกัน

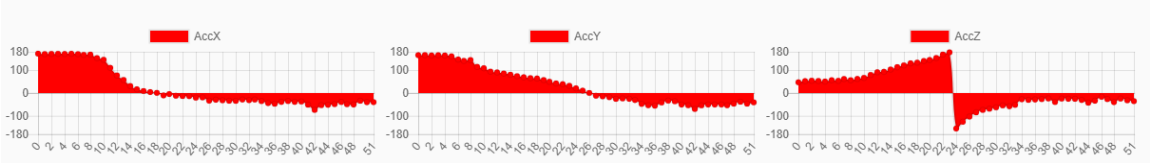
4.1.2 การทดลองใช้ถุงมือแบบที่ 2

4.1.2.1 วิธีการทดลอง

ประดิษฐ์ถุงมือแบบที่ 2 ขึ้นตามที่ออกแบบไว้ จากนั้นลองอ่านข้อมูลที่ได้จากเซนเซอร์ทุกตัว (IMU 6 ตัว) สำหรับ IMU ทุกตัว จะอ่านเพียงแค่ค่าดิบของ Accelerometer 3 แกน และ Gyroscope 3 แกน ทั้งสองมีค่าตั้งแต่ -2000 ถึง 2000 แล้วนำมาคำนวณตามสูตรคำนวณหามุมจาก Accelerometer และ Gyroscope ให้กลายเป็นค่ามุม 3 ค่า โดย IMU ที่อยู่หลังมือจะมีค่ามุมระหว่าง 0 ถึง 360 และ IMU ที่อยู่ปลายนิ้วจะนำไปหาความแตกต่างจาก IMU ที่อยู่หลังมือก่อนกลายเป็นค่าความแตกต่างระหว่างช่วง -180 ถึง 180 จากนั้นลองทำท่านิ่งเบื้องต้น ได้แก่ท่ากำมือ แบมือ ชูนิ้วแต่ละนิ้ว ชูนิ้วชี้กับนิ้วกลางพร้อมกัน และไขว้นิ้ว

4.1.1.2 ผลการทดลอง

ถุงมือไม่มีปัญหาด้านการจับค่า ท่าที่เคยมีปัญหาในถุงมือแบบที่ 1 มีค่าแตกต่างกันอย่างชัดเจน ทำให้สามารถแยกท่าที่เคยมีปัญหาออกจากกันได้



รูป 4.2 แผนภาพแสดงชุดข้อมูลเฉพาะนิ้วชี้ของท่าแบมือ (ถุงมือแบบที่ 2)

4.2 การทดลองเกี่ยวกับ Machine Learning

คือการทดลองเกี่ยวกับโมเดล Machine Learning ต่าง ๆ ที่ใช้ในการวิเคราะห์ท่าทางมือ โดยการทดลองอัลกอริธึมการสร้างโมเดลที่แตกต่างกัน

ท่าที่ใช้ในการทดสอบจะแตกต่างกันไประหว่างโมเดล มีทั้งท่านิ่งและท่าเคลื่อนไหว โดยท่าทั้งหมดที่ใช้ในการทดสอบจะมีดังนี้

ก) ข) ค)

ง) จ) ฉ)

รูป 4.3 ท่านิ่งที่ใช้ในการทดสอบ

ก) กำมือ

ข) แบมือ

ค) บอกรัก

ง) ชูสองนิ้ว

จ) ไขว้นิ้ว

ฉ) พยัญชนะ ก

ก) ข)

รูป 4.4 ท่าเคลื่อนไหวที่ใช้ในการทดสอบ - พยัญชนะ ข

ก) ขั้นตอนแรก

ข) ขั้นตอนที่สอง

ก) ข) ค)

รูป 4.5 ท่าเคลื่อนไหว - พยัญชนะ ค

ก) ขั้นตอนแรก

ข) ขั้นตอนที่สอง

ค) ขั้นตอนที่สาม

ก) ข)

รูป 4.6 ท่าเคลื่อนไหว - พยัญชนะ ง

ก) ขั้นตอนแรก

ข) ขั้นตอนที่สอง

4.2.1 การทดลองโมเดลแบบที่ 1: การฝึกโมเดล

4.2.1.1 วิธีการทดลอง

อ่านค่าชุดข้อมูล (Dataset) ที่ได้จากการอัดในส่วนของเว็บแอพพลิเคชั่น แบ่งข้อมูลที่จะนำเข้าไปฝึกโดยใช้ข้อมูลร้อยละ 70 ของทั้งหมดในการฝึก ส่วนที่เหลือใช้ในการประเมินผลลัพธ์ของโมเดล (Evaluate) นำข้อมูลที่จะฝึกไปเปลี่ยนรูป (Transform) ให้ข้อมูลอยู่ในช่วงเดียวกัน หลังจากนั้นสร้างโมเดลขึ้นมา โดยใช้จำนวนครั้งในการฝึก (Epochs) 10,000 ครั้ง จึงนำโมเดลไปประเมินด้วยข้อมูลที่ไม่ได้นำไปฝึก

ถุงมือที่ใช้ในโมเดลนี้คือถุงมือแบบที่ 1 ท่าที่ใช้ในการฝึกและทดสอบโมเดลนี้ได้แก่ กำมือ, แบมือ, บอกรัก, ชูสองนิ้ว และไขว้นิ้ว

4.2.1.2 ผลการทดลอง

โมเดลที่ได้มีความแม่นยำสูงมาก ผลจากการประเมินสูงเกือบถึงร้อยละ 100 แต่ไม่สามารถทำนายท่าทางเคลื่อนไหวได้

4.2.2 การทดลองโมเดลแบบที่ 1:ทดสอบความแม่นยำ

4.2.2.1 วิธีการทดลอง

นำโมเดลที่ได้จากขั้นตอนการฝึกอัพโหลดลงไปยังฝั่งเซิร์ฟเวอร์เพื่อทดสอบความแม่นยำ โดยการลองทำท่านิ่งทั้งหมด 50 ครั้ง ว่าผลลัพธ์ถูกต้องตามที่ต้องการหรือไม่

4.2.2.2 ผลการทดลอง

จากการนำไปทำนายท่านิ่งแต่ละท่า ทั้งหมด 50 ครั้ง ถูกต้อง 41 ครั้งหรือค่าความแม่นยำร้อยละ 82 เนื่องจากมีบางท่าที่เป็นข้อจำกัดของถุงมือแบบที่ 1 ทำให้การทำนายเกิดข้อผิดพลาด

ตาราง 4.1 ผลการทดสอบความแม่นยำโมเดลแบบที่ 1

ชื่อท่า	จำนวนครั้งที่ทดสอบ	จำนวนครั้งที่ถูก	ความแม่นยำ
กำมือ	10	10	100%
แบมือ	10	10	100%
บอกรัก	10	10	100%
ชูสองนิ้ว	10	10	100%
ไขว้นิ้ว	10	1	10%
รวม	50	41	82%

ตาราง 4.2 Confusion Matrix ของโมเดลแบบที่ 1

		ท่าจริง				
		กำมือ	แบมือ	บอกรัก	ชูสองนิ้ว	ไขว้นิ้ว
ท่าที่ตรวจจับได้	กำมือ	10	0	0	0	0
	แบมือ	0	10	0	0	0
	บอกรัก	0	0	10	0	0
	ชูสองนิ้ว	0	0	0	10	9
	ไขว้นิ้ว	0	0	0	0	1

4.2.3 การทดลองโมเดลแบบที่ 2: การฝึกโมเดล

4.2.3.1 วิธีการทดลอง

อ่านค่าชุดข้อมูล (Dataset) ที่ได้จากการอัดส่วนของเว็บแอพพลิเคชั่น แบ่งข้อมูลที่นำเข้าไปฝึกเป็นร้อยละ 70 นำข้อมูลไปเปลี่ยนรูป หลังจากนั้นสร้างโมเดลขึ้นมาโดยเปลี่ยนชั้นซ่อน (Hidden layer) เรื่อยๆ จนเจอค่าเหมาะสม ที่จะทำให้โมเดลสมบูรณ์มากขึ้น ปรับเปลี่ยนโมเดลไม่ให้ค่าฟิตเกินไปโดยใช้ Dropout ใช้จำนวนครั้งในการฝึกประมาณ 150 ครั้ง จึงนำโมเดลไปประเมิน

ถุงมือที่ใช้ในโมเดลนี้คือถุงมือแบบที่ 2 ท่าที่ใช้ในการฝึกและทดสอบโมเดลนี้ได้แก่ กำมือ, แบมือ, บอกรัก, ชูสองนิ้ว และไขว้นิ้ว

4.2.3.2 ผลการทดลอง

โมเดลที่ได้มีความแม่นยำสูง ผลจากการประเมินสูงเกือบร้อยละ 100 เช่นเดียวกับโมเดลแบบแรก ไม่สามารถทำนายท่าเคลื่อนไหวได้เช่นเดียวกัน

4.2.4 การทดลองโมเดลแบบที่ 2: ทดสอบความแม่นยำ

4.2.4.1 วิธีการทดลอง

นำโมเดลที่ได้อัพโหลดลงไปยังฝั่งเซิฟเวอร์ เพื่อทำสอบความแม่นยำ โดยลองทำท่านิ่งทั้งหมด 50 ครั้ง ว่าผลลัพธ์ถูกต้องตามที่ต้องการหรือไม่

4.2.4.2 ผลการทดลอง

จากการนำไปทำนายท่านิ่งในแต่ละท่า ทั้งหมด 50 ครั้ง ถูกต้อง 48 ครั้งหรือค่าความแม่นยำร้อยละ 96 เนื่องจากการใช้ถุงมือแบบที่ 2 ลดข้อจำกัดด้านการงอนิ้วลง ทำให้ผลลัพธ์ที่ได้แม่นยำมากขึ้น

ตาราง 4.3 ผลการทดสอบความแม่นยำโมเดลแบบที่ 2

ชื่อท่า	จำนวนครั้งที่ทดสอบ	จำนวนครั้งที่ถูก	ความแม่นยำ
กำมือ	10	10	100%
แบมือ	10	10	100%
บอกรัก	10	10	100%
ชูสองนิ้ว	10	10	100%
ไขว้นิ้ว	10	8	80%
รวม	50	48	96%

ตาราง 4.4 Confusion Matrix ของโมเดลแบบที่ 2

		ท่าจริง				
		กำมือ	แบมือ	บอกรัก	ชูสองนิ้ว	ไขว้นิ้ว
ท่าที่ตรวจจับได้	กำมือ	10	0	0	0	0
	แบมือ	0	10	0	0	0
	บอกรัก	0	0	10	0	0
	ชูสองนิ้ว	0	0	0	10	2
	ไขว้นิ้ว	0	0	0	0	8

4.2.5 การทดลองโมเดลแบบที่ 3: การฝึกโมเดล

4.2.5.1 วิธีการทดลอง

อ่านค่าชุดข้อมูล (Dataset) ที่ได้จากการอัดส่วนของเว็บแอพพลิเคชั่น ทำการแบ่งกลุ่มของข้อมูลเป็นลำดับของเวลา โดยใช้ทั้งหมด 20 ลำดับของค่ามุมในทุก ๆ นิ้ว เพื่อวิเคราะห์ข้อมูลท่าทางทั้งหมดที่แสดงในช่วงเวลาหนึ่ง ใช้ Dropout ในการปรับไม่ให้โมเดลฟิตเกินไป จำนวนครั้งในการฝึกประมาณ 100 ครั้ง

ถุงมือที่ใช้ในโมเดลนี้คือถุงมือแบบที่ 2 ท่าที่ใช้ในการฝึกและทดสอบโมเดลนี้ได้แก่ กำมือ, แบมือ, ชูสองนิ้ว, ไขว้นิ้ว, พยัญชนะ ก, ข, ค และ ง

4.2.5.1 ผลการทดลอง

โมเดลที่ได้มีความแม่นยำ แต่เนื่องจากเป็นโมเดลที่ประมวลผลลำดับด้วยทำให้โมเดลมีความซับซ้อน ใช้เวลาในการฝึกค่อนข้างนานและจำเป็นต้องใช้ชุดข้อมูลจำนวนมากเพื่อให้ได้ผลลัพธ์ในทุกๆกรณีที่จะเกิด

4.2.6 การทดลองโมเดลแบบที่ 3: ทดสอบความแม่นยำ

4.2.6.1 วิธีการทดลอง

นำโมเดลที่ได้อัพโหลดลงไปยังฝั่งเซิฟเวอร์ เพื่อทดสอบความแม่นยำ โดยลองทำท่านิ่งและเคลื่อนไหวทั้งหมด 50 ครั้ง ว่าผลลัพธ์ถูกต้องตามที่ต้องการหรือไม่

4.2.6.2 ผลการทดลอง

จากการนำไปทำนายท่านิ่งและท่าเคลื่อนไหวในแต่ละท่า ทั้งหมด 80 ครั้ง ถูกต้อง 64 ครั้ง หรือความแม่นยำร้อยละ 80 เนื่องจากมีท่าเคลื่อนไหวทำให้เกิดข้อผิดพลาดมากที่สุดใน 3 โมเดล

ตาราง 4.5 ผลการทดสอบความแม่นยำโมเดลแบบที่ 3

ชื่อท่า	จำนวนครั้งที่ทดสอบ	จำนวนครั้งที่ถูก	ความแม่นยำ
กำมือ	10	10	100%
แบมือ	10	10	100%
ชูสองนิ้ว	10	10	100%
ไขว้นิ้ว	10	8	80%
ก	10	8	80%
ข	10	5	50%
ค	10	5	50%
ง	10	8	80%
รวม	50	64	80%

ตาราง 4.6 Confusion Matrix ของโมเดลแบบที่ 3

		ท่าจริง								
		กำมือ	แบมือ	บอกรัก	ชูสองนิ้ว	ไขว้นิ้ว	ก	ข	ค	ง
ท่าที่ตรวจจับได้	กำมือ	10	0	0	0	0	0	0	0	1
	แบมือ	0	10	0	0	0	0	0	0	0
	บอกรัก	0	0	10	0	0	0	0	0	0
	ชูสองนิ้ว	0	0	0	10	1	1	0	0	0
	ไขว้นิ้ว	0	0	0	0	8	0	5	5	0
	ก	0	0	0	0	0	8	0	0	0
	ข	0	0	0	0	0	0	5	0	0
	ค	0	0	0	0	0	0	0	5	1
	ง	0	0	0	0	0	0	0	0	8

4.3 การทดลองเกี่ยวกับเว็บแอพพลิเคชั่น

คือการทดลองพัฒนาเว็บแอพพลิเคชั่นในส่วนต่าง ๆ รวมถึงการเชื่อมต่อ แลกเปลี่ยนข้อมูลกับส่วนอื่น ๆ เพื่อประยุกต์ใช้งานในขั้นตอนต่าง ๆ ของโครงงาน

4.3.1 การทดลองการส่งค่าผ่าน Socket.IO

4.3.1.1 วิธีการทดลอง

เชื่อมต่อเว็บแอพพลิเคชั่นเข้ากับเซิร์ฟเวอร์ผ่านทาง Socket.IO เมื่อเริ่มเข้าเว็บไซต์ ส่งข้อมูลจำนวน 20 ชุดต่อวินาทีจากเซิร์ฟเวอร์ไปยังเว็บแอพพลิเคชั่น จากนั้นลองกดปุ่มบนเว็บแอพพลิเคชั่นเพื่อส่งข้อมูลล่าสุดทั้ง 20 ชุดกลับไปวิเคราะห์ที่เซิร์ฟเวอร์

4.3.1.2 ผลการทดลอง

ทั้งสองฝั่งสามารถส่งข้อมูลหากันได้ทุกครั้งตลอดกระบวนการ

4.3.2 การทดลองการแสดงข้อมูลแบบเปลี่ยนแปลงตามเวลา

4.3.2.1 วิธีการทดลอง

สร้างแผนภาพเส้นแสดงค่า 20 ค่าล่าสุดจากเซนเซอร์แต่ละค่าที่ส่งมาจากเซิร์ฟเวอร์

4.3.2.2 ผลการทดลอง

สามารถแสดงค่าได้ตรงตามที่อ่านได้จากเซิร์ฟเวอร์

4.3.3 การทดลองบันทึกชุดข้อมูลของท่าต่าง ๆ และส่งออกเป็นไฟล์ .CSV

4.3.3.1 วิธีการทดลอง

ในหน้านักพัฒนา สำหรับการบันทึกรายครั้ง ตั้งค่าชื่อท่าแล้วกดปุ่ม RECORD เพื่อเริ่มบันทึกค่าที่ส่งมาจากเซิร์ฟเวอร์ ระหว่างบันทึกให้ทำท่าตามที่ต้องการ กดปุ่ม Stop เพื่อหยุดบันทึก จากนั้นกดปุ่ม Save เพื่อบันทึกท่า ทำแบบนี้จนครบ 50 ครั้ง เพื่อให้ได้ชุดข้อมูลจำนวนที่ขั้นต่ำสำหรับหนึ่งท่า

สำหรับการบันทึกต่อเนื่อง กดปุ่ม AUTO เพื่อเริ่มบันทึกค่าที่ส่งมาจากเซิร์ฟเวอร์ทุก ๆ 20 ค่า ค่าถัดไปจะถูกตัดเป็นชุดข้อมูลใหม่ ระหว่างบันทึกให้ทำท่าตามที่ต้องการ ทำจนครบอย่างน้อย 50 ครั้ง กดปุ่ม Stop เพื่อหยุดบันทึก จากนั้นกดปุ่ม Save เพื่อบันทึกท่า เพื่อให้ได้ชุดข้อมูลจำนวนที่ขั้นต่ำสำหรับหนึ่งท่า

จากนั้นกดปุ่ม Export เพื่อส่งออกชุดข้อมูลทั้งหมดออกเป็นไฟล์ export.csv

4.3.3.2 ผลการทดลอง

ทั้งการบันทึกรายครั้งและบันทึกต่อเนื่อง สามารถบันทึกค่าลงเป็นไฟล์ csv และนำไปใช้ในกระบวนการฝึกโมเดล Machine Learning ได้อย่างไม่มีปัญหา

บรรณานุกรม

Rung-Huei Liang, Ming Ouhyoung. “A real-time continuous gesture recognition system for sign \ Language”. [Online] Available: https://ieeexplore.ieee.org/abstract/document/671007/

Ji-Hwan Kim, Nguyen Duc Thang, Tae-Seong Kim. “3-D hand motion tracking and gesture recognition using a data glove”. [Online] Available: https://ieeexplore.ieee.org/abstract/document/5221998/

Felipe A. Quirino, Marcelo Romanssini, Rafael R. Dorneles, Enzo H. Weber, and Alessandro Girardi. “A Gesture Detection Glove for Human-Computer Interaction”. [Online] Available: http://ieee-cas.org/sites/ieee-cas.org/files/2017-2018-final-report_r9_a-gesture-detection-glove.pdf

NanoPI-DUO Hardware spec. [Online] Available: http://wiki.friendlyarm.com/wiki/index.php/NanoPi_Duo#Hardware_Spec

LSTM . [Online] Available: https://medium.com/@tongkornkitt/ml-lstms-%E0%B9%81%E0%B8%9A%E0%B8%9A-%E0%B9%80%E0%B8%81%E0%B8%B7%E0%B8%AD%E0%B8%9A-%E0%B8%A5%E0%B8%B0%E0%B9%80%E0%B8%AD%E0%B8%B5%E0%B8%A2%E0%B8%94%E0%B8%A2%E0%B8%B4%E0%B8%9A-%E0%B9%86-a3a55cd37883